Sequential Nonlinear Learning for Distributed Multiagent Systems via Extreme Learning Machines

Nuri Denizcan Vanli, Muhammed O. Sayin, Ibrahim Delibalta, and Suleyman Serdar Kozat, Senior Member, IEEE

Abstract—We study online nonlinear learning over distributed multilayer systems, where each agent employs a single hidden layer feedforward neural network (SLFN) structure to sequentially minimize arbitrary loss functions. In particular, each agent trains its own SLFN using only the data that is revealed to itself. On the other hand, the aim of the multilayer system is to train the SLFN at each agent as well as the optimal centralized batch SLFN that has access to all the data, by exchanging information between neighboring agents. We address this problem by introducing a distributed subgradient-based extreme learning machine algorithm. The proposed algorithm provides guaranteed upper bounds on the performance of the SLFN at each agent and shows that each of these individual SLFNs asymptotically achieves the performance of the optimal centralized batch SLFN. Our performance guarantees explicitly distinguish the effects of data- and network-dependent parameters on the convergence rate of the proposed algorithm. The experimental results illustrate that the proposed algorithm achieves the oracle performance significantly faster than the state-of-the-art methods in the machine learning and signal processing literature. Hence, the proposed method is highly appealing for the applications involving big data.

Index Terms—Distributed systems, extreme learning machine (ELM), multiagent optimization, sequential learning, single hidden layer feedforward neural networks (SLFNs).

I. INTRODUCTION

A. Preliminaries

The demand for neural network inspired learning structures is steadily growing owing to their superior non-linear modeling power and low complexity [1]–[3]. Although several neural-adaptive learning methods [1]–[5] are used for processing data in a centralized manner, the steadily increasing growth of the data sizes (in terms of both dimensionality and length) prohibits centralized processing due to computational complexity, storage, and communication issues [6], [7].

Manuscript received April 30, 2015; accepted February 26, 2016. Date of publication March 11, 2016; date of current version February 15, 2017. This work was supported in part by the Turkish Academy of Sciences Outstanding Researcher Programme within the Scientific and Technological Research Council of Turkey under Contract 113ES17 and Contract 115ES17 and in part by Türk Telekom Laboratories.

N. D. Vanli, M. O. Sayin, and S. S. Kozat are with the Department of Electrical and Electronics Engineering, Bilkent University, Ankara 06800, Turkey (e-mail: vanli@ee.bilkent.edu.tr; sayin@ee.bilkent.edu.tr; kozat@ee.bilkent.edu.tr).

I. Delibalta is with AveaLabs, Avea Communications Services Inc., Istanbul 34367, Turkey, and also with the Graduate School of Social Sciences and Humanities, Koç University, Istanbul 34450, Turkey (e-mail: ibrahim.delibalta@avea.com.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2536649

To address this problem, several distributed learning algorithms are proposed in the machine learning and signal processing literature [8]–[13]. Although these algorithms are shown to achieve certain statistical and deterministic convergence rates, they are usually based on linear models, which significantly limit their performance in real-life applications [14].

In this paper, we resolve these issues by introducing a sequential nonlinear kernel-adaptive learning algorithm with guaranteed convergence bounds without any statistical assumptions. In particular, we propose a distributed single hidden layer feedforward neural network (SLFN) structure with strong theoretical convergence guarantees in a deterministic sense, i.e., without any statistical assumptions on the data. Our algorithm builds upon the existing centralized extreme learning machine (ELM)-based methods in a novel manner by: 1) providing a rigorous distributed formulation of the neural network-based optimization problem and 2) introducing a low-complexity algorithm to achieve the performance of the optimal centralized ELM-based method. Through extensive set of simulations and real-life experiments, we demonstrate significant performance gains achieved by the proposed algorithm with respect to the conventional distributed learning algorithms in the signal processing and machine learning literature.

B. Prior Art and Comparison

For applications involving high levels of complexity and nonlinearity, several SLFN-based algorithms are introduced for accurate nonlinear modeling [1], [15]–[18]. In order to train SLFNs, a popular method is to use the ELM algorithm [1]. In particular, Huang et al. [1] illustrate the sufficient initialization conditions to train the SLFNs, which significantly improve the convergence speed of the SLFN-based algorithms. Liang et al. [2] have extended these ideas to sequential learning problems. Recently, several variants of the ELM method are proposed for various learning tasks and cost functions. Specifically, Balasundaram et al. [5] considered the one-norm minimization and introduced a method to generalize the ELM algorithm for this cost measure. In [19], the ELM algorithm is used for ranking problems, and in [20], hierarchical structures are incorporated into the ELM framework to process high-dimensional data. Our work builds upon these existing works in the literature by providing a scalable distributed method, over which any such ELM-based algorithm can be applied. Furthermore, the computational complexity of the proposed algorithm is linear in the number
of hidden nodes for each agent, whereas it is quadratic for the original ELM method [2]. Therefore, the proposed algorithm is highly appealing for applications involving big data, since the introduced algorithm directly decreases: 1) the amount of data to be processed at each agent and 2) the computational complexity of the processing algorithms at each agent.

Distributed learning and optimization problems are extensively studied in the context of signal processing [8]–[10] and optimization [11]–[13], [21], [22] due to their natural modeling abilities of underlying phenomenon. In the distributed setting, the authors use linear models to capture the relationship between the input and output data [8]–[10]. However, the convergence results in these papers are only valid under certain statistical assumptions on the data, which do not usually hold in real-life applications. In this paper, we do not commit to linear models, but instead use an SLFN-based structure, which has shown to elegantly model the complex nonlinear relationships [1]. Furthermore, our analyses hold in an individual sequence manner without any statistical assumptions on the data, unlike [8]–[10].

We emphasize that the conventional distributed multiagent optimization algorithms [11]–[13], [21], [22] usually have linear structures in general, similar to [8]–[10]. On the other hand, these methods are mainly based on subgradient methods, and the theoretical analyses are carried out from a deterministic perspective. Yet, these subgradient methods may not provide satisfactory performance if the cost function is nondifferentiable at certain points. This holds, since these nondifferentiable points are usually the desired solutions of the optimization problems (e.g., consider the regularized one-norm loss). Therefore, in this paper, we do not directly use a subgradient-based approach, but instead use a splitting method [23]–[25] to ensure that our algorithm attains such nondifferentiable values. In this manner, our algorithm achieves a considerably smaller accumulated error and illustrates a better convergence performance as shown in our experiments.

C. Contributions

Our main contributions are as follows.

1) We introduce a sequential nonlinear optimization algorithm over distributed multiagent learning systems. Here, the multiagent structure optimizes the SLFNs for both additive and radial basis function kernels in a fully distributed manner. The proposed algorithm is truly sequential, such that it processes each new data pair and update the SLFN model without the knowledge of the time horizon.

2) We show that by our diffusion scheme, each agent can successfully and uniformly optimize the SLFN weights to minimize the overall network cost (over the entire data) with observing only a portion of the data. We demonstrate this result in a deterministic sense without any statistical assumptions on the data, such that our results are guaranteed to hold uniformly for all input and output sequences.

3) We achieve this performance with a computational complexity only linear in the data length. Thus, our algorithm can be efficiently used in applications involving big data.

4) We demonstrate the significant performance gains achieved by our algorithm over numerical examples and benchmark real data sets.

D. Organization of This Paper

The organization of this paper is as follows. In Section II, we introduce the multiagent learning problem and use the SLFNs as well as the forward–backward splitting method to derive a sequential distributed optimization algorithm. We then analyze the convergence performance of the introduced algorithm and provide guaranteed regret bounds in Sections III and IV. In Section V, we compare the performance of our algorithm with respect to the state-of-the-art algorithms over benchmark data sets. This paper concludes with several remarks in Section VI.

II. Problem Description

A. Notation

Throughout this paper, all vectors are column vectors and represented by boldface lowercase letters. Matrices are represented by boldface uppercase letters. For a matrix \( H \), \( ||H||_F \) is the Frobenius norm. For a vector \( x \) (and matrix \( H \)), \( ||x|| \) (and \( ||H|| \)) is the \( l^2 \)-norm. For two vectors \( x, y \in \mathbb{R}^n \), \( \langle x, y \rangle = x^T y \) is the inner product. Here, \( 0 \) (and \( 1 \)) denotes the vector with all zeros (and ones), and the dimensions can be understood from the context. For a matrix \( H, H_{jk} \) represents its entry at the \( j \)th row and the \( k \)th column.

B. System Overview

We study the distributed sequential training of SLFN structures, which can be used in various applications, including nonlinear optimization, regression, and classification. As an example, in wireless sensor networks, a number of agents observe different data sequences related to (or generated by) a phenomenon of interest. In a centralized approach to this problem (see [2] and references therein), the entire data sequences are required to be processed by a single centralized agent. To this end, each agent transmits its observations into a centralized processor, where a sequential algorithm, e.g., online ELM [2] is applied. However, this centralized structure cannot process the data in a truly sequential manner, since the data are delayed due to transmission and then processed in chunks at the centralized processor. Furthermore, each agent may have the capability of generating huge volumes of data due to the recent developments in hardware technologies [8]. Therefore, centralized processing of such huge amounts of data may not be feasible or even possible due to the transmission, computational complexity, and storage issues [10].

To resolve these problems, we introduce a distributed framework, where each agent sequentially processes its own data and shares the extracted information with its neighbors, as shown in Fig. 1. Thus, the data are processed in a truly sequential and completely decentralized manner. Nevertheless, as rigorously shown in Section IV, the proposed distributed sequential method achieves asymptotically the same performance with its optimal centralized batch variant.
can rewrite (1) in the vector form as

$$\hat{y}_{t,k} = \langle w_{t,k}, z_{t,k} \rangle$$

where $z_{t,k} \triangleq [z(a_{1,k}, b_{1,k}; x_{t,k}), \ldots, z(a_{N,k}, b_{N,k}; x_{t,k})]^T$ and $w_{t,k} \triangleq [w_{t,1,k}, \ldots, w_{t,N,k}]^T$.

In the distributed setting, the aim of each agent is to minimize a cost function $f_{t,k}() + r_k()$ at time $t$ over a convex set $\mathcal{W}$, where $f_{t,k}()$ and $r_k()$ are the generic convex functions. Here, for the $k$th agent, the function $f_{t,k}()$ is an empirical loss, and the function $r_k()$ is a regularization term. As an example, by setting $f_{t,k}(w) = \langle z_t - (w, z_{t,k}) \rangle^2$ and $r_k(w) = \lambda ||w||_p$, we obtain the regularized least squares cost function. We emphasize that we consider this generic cost function, since we can train the ELMs using different cost functions instead of the squared error loss depending on the application. Our approach covers these different cases and models a broad range of practical cost functions that are widely used in optimization [32], machine learning [33], and signal processing literature [34]. In a more formal manner, the aim of each agent is to sequentially solve the following optimization problem:

$$\min_{i=1}^{T} \sum_{i=1}^{T} [f_{i,k}(w_{t,k}) + r_k(w_{t,k})]$$

s.t. $w_{t,k} \in \mathcal{W}$

for any $T$, where $T$ is the length of the data, which is arbitrary and unknown, i.e., the agents perform optimization in a truly sequential manner without the knowledge of the time horizon.

Although the objective of each agent is to minimize the cost function in (3), the aim in the distributed framework is to propose an algorithm that achieves asymptotically the same performance as the optimal batch centralized processor. To this end, we define the following global loss functions:

$$f_i(w_{t,k}) \triangleq \sum_{j=1}^{K} f_{i,j}(w_{t,k})$$

$$r(w_{t,k}) \triangleq \sum_{j=1}^{K} r_j(w_{t,k})$$

which represent the loss of the SLFN weight vector of the $k$th agent on the data observed by all agents on the distributed network at time $t$. In this way, we aim to construct a distributed learning algorithm that is consistent among all the agents, i.e., each agent should be able to acquire the information from every input and output data pairs observed by every other agent. In particular, our aim is to solve the following optimization problem:

$$\min_{w_{t,k} \in \mathcal{W}} \sum_{i=1}^{T} [f_i(w_{t,k}) + r(w_{t,k})]$$

over all $k \in \{1, \ldots, K\}$. Here, we emphasize that we avoid any statistical assumptions on the input vector and the target data. Instead, our aim is to introduce a sequential algorithm that provides guaranteed performance in a strong deterministic sense. To this end, we next introduce an algorithm, which is used by each agent to minimize the cost function of that
agent (5), (6). Then, SLFN weight vectors trained by this algorithm are diffused among the agents to solve the global SLFN-based optimization problem in (7).

C. Forward–Backward Splitting Method

For the applications involving big data and/or requiring sequential processing, the dimensionality of the input data can be relatively large [14]. Hence, we aim to design highly efficient algorithms with low complexity to solve the optimization problem in (7). To this end, we use subgradient methods [11] due to their computational efficiency and generalization capabilities [12]. The set of subgradients of a function $i$ at $w$ is defined as follows:

$$\hat{\partial}i(w) = \{g \in \mathbb{R}^m : i(v) \geq i(w) + \langle g, v-w \rangle \ \forall v \in \mathbb{R}^m\}. \quad (8)$$

By using the subgradient method to minimize (5) at each individual node $k$, we obtain iterates of the form $w_{t+1,k} = w_{t,k} - \mu_k g_{t,k} - \mu_t h_{t,k}$, where $\mu_k$ is the learning rate of the subgradient method, $g_{t,k} \in \hat{\partial}f_{t,k}(w_t)$, and $h_{t,k} \in \hat{\partial}r_k(w_t)$. However, if the cost functions $f_{t,k}(\cdot)$ or $r_k(\cdot)$ are nondifferentiable at certain points, then the iterates $w_{t,k}$ cannot usually attain these values [32], which are usually the desired minima of the cost functions. As an example, for $r_k(w) = \lambda \|w\|_1$, the desired minima is rarely achievable by the aforementioned update rule. We emphasize that such nondifferentiable cost functions are usually studied in the context of optimization and learning theory via the SLFNs and ELM methods [2].

To address this problem, we use the forward–backward splitting method [25]. In particular, without the loss of generality, we assume that the cost function $f_{t,k}(\cdot)$ is differentiable, and our aim is to ensure that the iterates $w_{t,k}$ attain the nondifferentiable points of the function $r_k(\cdot)$ (note that if $f_{t,k}(\cdot)$ is nondifferentiable at some points, whereas $r_k(\cdot)$ is differentiable, one can easily switch the labels of the cost functions and apply the below procedure). To solve the optimization problem in (5), we perform the following updates:

$$w_{t+1,k} = w_{t,k} - \mu_k g_{t,k} \quad (9)$$

$$w_{t+1,k} = \arg\min_w \left\{ \frac{1}{2} \|w - w_{t+1,k}\|^2 + \mu_t r_k(w) \right\} \quad (10)$$

where (9) updates the parameter vector using a subgradient of the first function $f_{t,k}(\cdot)$. Then, in (10), we seek to minimize the regularization cost $r_k(\cdot)$ while not getting too far from the intermediate parameter vector calculated in (9). Note that by using the updates in (9) and (10), we make sure that the iterates attain the aforementioned nondifferentiable points due to the following observation. The zero vector should be an element of the set of subgradients of (10). Therefore, $0 \in w_{t+1,k} - w_{t+1,k} + \mu \partial r_k(w_{t+1,k})$. This indicates that there always exists a subgradient vector $h_{t+1,k} \in \hat{\partial}r_k(w_{t+1,k})$, such that

$$w_{t+1,k} = w_{t,k} - \mu_k g_{t,k} - \mu_t h_{t,k}. \quad (11)$$

That is, we can reach any nondifferentiable points.

In the following, we introduce a distributed algorithm that uses the subgradient method described in this section to train the SFLN models in a truly sequential manner over distributed architectures and derive the corresponding guaranteed performance bounds.

III. DISTRIBUTED SEQUENTIAL SPLITTING EXTREME LEARNING MACHINE

In this section, we introduce a sequential distributed SLFN-based algorithm, which uses the forward–backward splitting method for optimization (the complete description of the algorithm is given in Algorithm 1). At each time $t$, each agent $k$ calculates its estimate of the target data as in (2) and then diffuses the weights of the SFLN, i.e., $w_{t,k}$, to its neighbors, as shown in Fig. 1. Each agent $k$, after receiving the weight vectors from the neighboring nodes, performs a weighted averaging

$$\phi_{t+1,k} = \sum_{j=1}^{K} C_{jk} w_{t,j} \quad (12)$$

to construct an intermediate weight vector, where $C$ is the communication matrix of the graph, such that $C_{jk}$ are the combination weights. After diffusion of the weights of the SLFN, each agent then performs its update using the forward–backward splitting method based on the combined weight vector, as described in Section II-C.

We analyze the performance of this sequential distributed kernel-adaptive learning algorithm using the following standard assumptions.

1) The convex set $W$ has a diameter of $D$, i.e., $\|w - v\| \leq D, \forall w, v \in W$.
2) The norms of the subgradients are bounded by $A$, i.e., $\|g_{t,k}\|, \|h_{t,k}\| \leq A, \forall g_{t,k} \in \hat{\partial}f_{t,k}(w_t)$ and $\forall h_{t,k} \in \hat{\partial}r_k(w_t)$.
3) The communication graph $C$ forms a doubly stochastic matrix, such that $C$ is irreducible and aperiodic, i.e., $C^T = C = I$.
4) Initial weights are equal at each node, i.e., $w_{1,k} = w_{1,j}, \forall k, j \in \{1, \ldots, K\}$.

Here, the first assumption enforces a bounded parameter vector, which is clearly realistic, since our cost function has regularization penalties; thus, any parameter vector of interest should be bounded. Similar diameter boundedness assumptions are also used in a number of different papers in the literature [22], [35]. The second assumption holds for a variety of loss functions, since our optimization space is bounded, and consequently, the outputs of the hidden nodes are also bounded. As an example, any cost function whose gradient is

---

**Algorithm 1 Distributed Sequential Splitting ELM**

1. for $t = 1$ to $T$ do
2. for $k = 1$ to $K$ do
3. $\hat{y}_{t,k} = (w_{t,k}, z_{t,k})$
4. $\phi_{t+1,k} = \sum_{j=1}^{K} C_{jk} w_{t,j}$
5. $w'_{t+1,k} = \phi_{t+1,k} - \mu_t g_{t,k}$
6. $w_{t+1,k} = \arg\min_w \frac{1}{2} \|w - w'_{t+1,k}\|^2 + \mu_t r_k(w)$
7. end for
8. end for
The proposed algorithm is truly sequential and consequently highly suitable for applications involving high-dimensional data.

For Algorithm 1, we have Theorem 1, which states that the introduced algorithm minimizes a sum of convex objective functions with a regret of $\mathcal{O}(K^{1.25} \sqrt{T})$, uniformly at each agent. We note that the convergence result in Theorem 1 uniformly holds for all agents, without any statistical assumptions on the data and without any knowledge of the time horizon, i.e., our algorithm does not need $T$ in hindsight. Hence, the proposed algorithm is truly sequential and consequently highly suitable for applications involving high-dimensional data.

**Theorem 1:** Under Assumptions 1–4, Algorithm 1, when applied to any input and output data sequences, achieves the following convergence guarantee:

$$
\sum_{t=1}^{T} [f_t(w_{t,k}) + r(w_{t,k})] - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} [f_t(w) + r(w)] 
\leq \mathcal{O}
\left(\frac{AD}{\alpha} K^{1.25} \sqrt{T}\right)
$$

for all $T$ and $\forall k \in \{1, \ldots, K\}$, with a suitable learning rate. Here, $\alpha$ is the spectral gap of the communication matrix $C$, $D$ is the diameter of the convex set of the optimization problem, and $A$ is an upper bound on the norms of the subgradients.

This theorem states that the weights of the SLPNs at every agent uniformly yield the same asymptotical normalized accumulated error as the optimal weights that can be chosen only in hindsight, i.e., after observing all the data in a batch mode and centralized manner. Here, we define the normalized accumulated error of the proposed algorithm as follows:

$$
\frac{1}{T} \sum_{t=1}^{T} [f_t(w_{t,k}) + r(w_{t,k})]
$$

and the normalized accumulated error of the competitor algorithm can be defined by replacing $w_{t,k}$ with $w$. Therefore, as $T \to \infty$, the error per instance of our algorithm converges to the error per instance of the optimal batch variant. Thus, although each agent observes a data of length $T$, all agents can still achieve the normalized performance of the centralized batch algorithm (which processes the entire data of length $KT$) through the communication between agents for any given network (with a finite size). Furthermore, if the cost function $f_\cdot(\cdot)$ is $F$-strongly convex and $r(\cdot)$ is $R$-strongly convex, where $S \triangleq F + R > 0$, then Algorithm 1 achieves a convergence guarantee of $\mathcal{O}(K^{1.5} \log T)$, as presented in Corollary 1.

**Corollary 1:** Under Assumptions 1–4 and when the sum of the cost functions are $S$-strongly convex for some $S > 0$, Algorithm 1, when applied to any input and output data sequences, achieves the following convergence guarantee:

$$
\sum_{t=1}^{T} [f_t(w_{t,k}) + r(w_{t,k})] - \min_{w \in \mathcal{W}} \sum_{t=1}^{T} [f_t(w) + r(w)] 
\leq \mathcal{O}
\left(\frac{A^2}{\alpha} K^{1.5} \log T\right)
$$

for all $T$ and $\forall k \in \{1, \ldots, K\}$, with a suitable learning rate. Here, $\alpha$ is the spectral gap of the communication matrix $C$ and $A$ is an upper bound on the norms of the subgradients.

**IV. PROOFS**

**A. Proof of Theorem 1**

In order to relate the performance of the weights of an individual agent to the optimal weights, we define an average weight vector. We first compare the performance of this weight vector with respect to the optimal weights and then relate this average weight to the weights at each agent. To this end, let

$$
\tilde{w}_t = \frac{1}{K} \sum_{k=1}^{K} w_{t,k}
$$

denote the average weight vector. According to this average weight, we also define the following subgradients:

$$
\tilde{g}_{t,k} \in \partial f_{t,k}(\tilde{w}_t) \quad (14)
$$

$$
\tilde{h}_{t,k} \in \partial r_{t,k}(\tilde{w}_t). \quad (15)
$$

We then consider the average estimation parameter $\tilde{w}_t$ in (13) and obtain the following recursion:

$$
\tilde{w}_{t+1} = \frac{1}{K} \sum_{k=1}^{K} w_{t+1,k}
$$

$$
= \frac{1}{K} \sum_{k=1}^{K} \left[ \sum_{j=1}^{K} C_{jk} w_{t,j} - \mu_{t} g_{t,k} - \mu_{t} h_{t+1,k} \right]
$$

$$
= \tilde{w}_t - \frac{\mu_{t}}{K} \sum_{k=1}^{K} (g_{t,k} + h_{t+1,k}) \quad (16)
$$

where the last line follows from Assumption 3, i.e., $\sum_{k=1}^{K} C_{jk} = 1$. By subtracting an arbitrary $w \in \mathcal{W}$ from both sides of (16) and taking the squared norm of both sides, we obtain

$$
||\tilde{w}_{t+1} - w||^2 = ||\tilde{w}_t - w||^2 + \frac{\mu_{t}^2}{K^2} \left[ \sum_{k=1}^{K} g_{t,k} + h_{t+1,k} \right]^2 - 2 \frac{\mu_{t}}{K} \sum_{k=1}^{K} (g_{t,k} + h_{t+1,k}) (\tilde{w}_t - w). \quad (17)
$$

The second term of (17) can be bounded as follows:

$$
\left[ \sum_{k=1}^{K} g_{t,k} + h_{t+1,k} \right]^2 \leq \left( \sum_{k=1}^{K} ||g_{t,k} + h_{t+1,k}|| \right)^2 \leq 4A^2 K^2. \quad (18)
$$
We then consider the third term of (17) and write the first inner product $-\langle g_{t,1,k}, \bar{w}_t - w \rangle$ as follows:

$$-\langle g_{t,1,k}, \bar{w}_t - w \rangle = -\langle g_{t,1,k}, \bar{w}_t - w_{t,1,k} \rangle - \langle g_{t,1,k}, w_{t,1,k} - w \rangle.$$  

(19)

The last term in (19) can be upper bounded using the convexity of $f_{t,k} (\cdot)$ as follows:

$$\langle g_{t,1,k}, w - w_{t,1,k} \rangle \leq f_{t,k} (w) - f_{t,k} (w_{t,1,k}) - \frac{F}{2} \| w - w_{t,1,k} \|^2$$

$$\leq f_{t,k} (w) - f_{t,k} (\bar{w}_t) + \langle \bar{g}_{t,k}, \bar{w}_t - w_{t,1,k} \rangle - \frac{F}{2} \| w - w_{t,1,k} \|^2 + \| w_{t,1,k} - \bar{w}_t \|^2).$$

(20)

Putting (20) back in (19) and noting that the norm of the gradients is upper bounded by $A$, we obtain

$$-\langle g_{t,1,k}, \bar{w}_t - w \rangle \leq f_{t,k} (w) - f_{t,k} (\bar{w}_t) + 2A \| \bar{w}_t - w_{t,1,k} \|$$

$$- \frac{F}{2} (\| w - w_{t,1,k} \|^2 + \| w_{t,1,k} - \bar{w}_t \|^2).$$

(21)

We next consider the second inner product in the third term of (17) and denote it as follows:

$$-\langle h_{t+1,1,k}, \bar{w}_t - w \rangle = -\left( h_{t+1,1,k} + \sum_{j=1}^{K} C_{j,k} w_{t,j} - w_{t+1,k} \right)$$

$$- \langle h_{t+1,1,k}, \bar{w}_t - \sum_{j=1}^{K} C_{j,k} w_{t,j} \rangle$$

$$- \langle h_{t+1,1,k}, w_{t+1,k} - w \rangle.$$  

(22)

Here, we observe that the first term of (22) can be upper bounded as

$$\left( h_{t+1,1,k}, w_{t+1,k} - \sum_{j=1}^{K} C_{j,k} w_{t,j} \right)$$

$$= (h_{t+1,1,k} - \mu_t g_{t,1,k} - \mu_t h_{t+1,k}) \leq 2A^2 \mu_t$$

(23)

where the second line follows from our update rule. We then consider the second term of (22) and upper bound it as

$$-\left( h_{t+1,1,k}, \bar{w}_t - \sum_{j=1}^{K} C_{j,k} w_{t,j} \right)$$

$$= -\left( h_{t+1,1,k}, \sum_{j=1}^{K} C_{j,k} (\bar{w}_t - w_{t,j}) \right)$$

$$\leq \sum_{j=1}^{K} C_{j,k} \| \bar{w}_t - w_{t,j} \|$$

(24)

where the second line follows due to Assumption 3, since $\sum_{j=1}^{K} C_{j,k} = 1$. Finally, we consider the third term of (22) and upper bound it using the convexity of $r_k (\cdot)$ [by using a similar trick that we have done in (20)] and obtain

$$\langle h_{t+1,1,k}, w - w_{t+1,k} \rangle$$

$$\leq r_k (w) - r_k (\bar{w}_{t+1}) + \langle h_{t+1,1,k}, \bar{w}_{t+1} - w_{t+1,k} \rangle - \frac{R}{2} (\| w - w_{t+1,k} \|^2 + \| w_{t+1,k} - \bar{w}_{t+1} \|^2).$$

(25)

Putting (23)–(25) back in (22), we obtain

$$-\langle h_{t+1,1,k}, \bar{w}_t - w \rangle$$

$$\leq r_k (w) - r_k (\bar{w}_{t+1}) + 2A^2 \mu_t$$

$$+ A \| \bar{w}_{t+1} - w_{t+1,k} \| + \sum_{j=1}^{K} C_{j,k} \| \bar{w}_t - w_{t,j} \|$$

$$- \frac{R}{2} (\| w - w_{t+1,k} \|^2 + \| w_{t+1,k} - \bar{w}_{t+1} \|^2).$$

(26)

Adding (21) and (26), and summing from $k = 1$ to $K$, we obtain the third term of (17) as follows:

$$- \sum_{k=1}^{K} \langle g_{t,1,k} + h_{t+1,1,k}, \bar{w}_t - w \rangle$$

$$\leq \sum_{k=1}^{K} \left( f_{t,k} (w) - f_{t,k} (\bar{w}_t) + r_k (w) - r_k (\bar{w}_{t+1}) + \sum_{j=1}^{K} C_{j,k} \| \bar{w}_t - w_{t,j} \|$$

$$+ 2A \| \bar{w}_t - w_{t,k} \| + A \| \bar{w}_{t+1} - w_{t+1,k} \|$$

$$+ 2A^2 \mu_t - \frac{F}{2} (\| w - w_{t,k} \|^2 + \| w_{t,k} - \bar{w}_t \|^2)$$

$$- \frac{R}{2} (\| w - w_{t+1,k} \|^2 + \| w_{t+1,k} - \bar{w}_{t+1} \|^2) \right).$$

(27)

From the definition of the subgradient, we have $f_{t,k} (\bar{w}_t) \geq f_{t,k} (w_{t,k}) + \langle g_{t,k}, w_{t,k} - \bar{w}_t \rangle$ and $r_k (\bar{w}_{t+1}) \geq r_k (w_{t+1,k}) + \langle h_{t+1,k}, w_{t+1,k} - \bar{w}_{t+1} \rangle$. Using these inequalities, we can write (27) as follows:

$$- \sum_{k=1}^{K} \langle g_{t,1,k} + h_{t+1,1,k}, \bar{w}_t - w \rangle$$

$$\leq \sum_{k=1}^{K} \left( f_{t,k} (w) - f_{t,k} (w_{t,k}) + r_k (w) - r_k (w_{t+1,k}) + \sum_{j=1}^{K} C_{j,k} \| \bar{w}_t - w_{t,j} \|$$

$$+ 3A \| \bar{w}_t - w_{t,k} \| + 2A\| \bar{w}_{t+1} - w_{t+1,k} \|$$

$$+ 2A^2 \mu_t - \frac{F}{2} (\| w - w_{t,k} \|^2 + \| w_{t,k} - \bar{w}_t \|^2)$$

$$- \frac{R}{2} (\| w - w_{t+1,k} \|^2 + \| w_{t+1,k} - \bar{w}_{t+1} \|^2) \right).$$

(28)

To simplify (28), we present Lemma 1, whose proof is given in Section IV-C.

**Lemma 1:** Under Assumptions 2–4, the deviation of the parameter vector of each agent from the average parameter vector is upper bounded as follows:

$$\| \bar{w}_t - w_{t,k} \| \leq 2A \sqrt{K} \sum_{s=1}^{l-1} \mu_{t-s} \sigma^{s-1}$$

(29)
where $\sigma$ is the second largest singular value of the communication matrix $C$.

Considering the definitions of the cost functions, noting that
\[
\frac{1}{K} \sum_{k=1}^{K} (\|w - w_{t,k}\|^2 + \|w_{t,k} - \bar{w}_{t}\|^2) \leq \|w - \bar{w}_{t}\|^2 \tag{30}
\]
from Jensen’s inequality, and using Lemma 1, we can rewrite (28) as follows:
\[
- \sum_{k=1}^{K} (g_{t,k} + h_{t+1,k}, \bar{w}_{t} - w) \\
\leq f_{t}(w) - f_{t}(w_{t,k}) \\
+ r(w) - r(w_{t+1,k}) + 8A^{2}K\sqrt{K} \sum_{s=1}^{t-1} \mu_{t-s}\sigma^{s-1} \\
+ 2A^{2}K\mu_{t} + 4A^{2}K\sqrt{K} \sum_{s=1}^{t} \mu_{t-s+1}\sigma^{s-1} \\
- \frac{KF}{2}(\|w - \bar{w}_{t}\|^2) - \frac{KR}{2}(\|w - \bar{w}_{t+1}\|^2). \tag{31}
\]

In order to obtain the desired upper bound, we put (18) and (31) back in (17) and summing from $t = 1$ to $T$, we obtain
\[
\sum_{t=1}^{T} f_{t}(w_{t,k}) - f_{t}(w) + r(w_{t+1,k}) - r(w) \\
\leq \sum_{t=1}^{T} \left\{ \frac{K}{2} \left( \frac{1}{\mu_{t} - F} \right) \|w - \bar{w}_{t}\|^2 \\
- \frac{K}{2} \left( \frac{1}{\mu_{t} + R} \right) \|w - \bar{w}_{t+1}\|^2 \\
+ 8A^{2}K\sqrt{K} \sum_{s=1}^{t-1} \mu_{t-s}\sigma^{s-1} \\
+ 4A^{2}K\mu_{t} + 4A^{2}K\sqrt{K} \sum_{s=1}^{t} \mu_{t-s+1}\sigma^{s-1} \right\}. \tag{32}
\]
Here, we make the following observation:
\[
\sum_{t=1}^{T} \sum_{s=1}^{t-1} \mu_{t-s}\sigma^{s-1} = \sum_{s=1}^{T} \sigma^{s-1} \sum_{t=s+1}^{T} \mu_{t-s} \\
\leq \frac{1}{1 - \sigma} \sum_{t=1}^{T} \mu_{t} \tag{33}
\]
which yields [by adding $r(\bar{w}_{1})$ to the both sides of the inequality] the following regret bound:
\[
\sum_{t=1}^{T} f_{t}(w_{t,k}) - f_{t}(w) + r(w_{t,k}) - r(w) \\
\leq \frac{K}{2} \sum_{t=2}^{T} \left( \frac{1}{\mu_{t}} - \frac{1}{\mu_{t-1}} - F - R \right) \|w - \bar{w}_{t}\|^2 \\
+ AD + K \left( \frac{1}{\mu_{2}} - F \right) \|w - \bar{w}_{1}\|^2 \\
+ 4A^{2}K \left( 1 + \frac{3\sqrt{K}}{1 - \sigma} \right) \sum_{t=1}^{T} \mu_{t}. \tag{34}
\]

If we have $F = R = 0$, i.e., if neither of the cost functions are strongly convex, we obtain the following upper bound:
\[
\sum_{t=1}^{T} f_{t}(w_{t,k}) - f_{t}(w) + r(w_{t,k}) - r(w) \\
\leq \frac{D^{2}K}{2} \left[ \frac{1}{\mu_{1}} + \sum_{t=2}^{T} \left( \frac{1}{\mu_{t}} - \frac{1}{\mu_{t-1}} - S \right) \right] \\
+ AD + 4A^{2}K \left( 1 + \frac{3\sqrt{K}}{1 - \sigma} \right) \sum_{t=1}^{T} \mu_{t}. \tag{35}
\]

Defining the function $\beta(K, \sigma) \triangleq 1 + (3\sqrt{K}/1 - \sigma)$ and letting $\mu_{t} = (D/4A(\beta(K, \sigma)t))^{1/2}$, we can rewrite the upper bound in (35) as follows:
\[
\sum_{t=1}^{T} f_{t}(w_{t,k}) - f_{t}(w) + r(w_{t,k}) - r(w) \\
\leq AD + 4A\sqrt{\beta(K, \sigma)} \tag{36}
\]
since
\[
\sum_{t=1}^{T} \frac{D}{4A\sqrt{\beta(K, \sigma)t}} \leq \frac{D\sqrt{T}}{2A}\sqrt{\beta(K, \sigma)}. \tag{37}
\]
This concludes the proof of Theorem 1. \hfill \Box

B. Proof of Corollary 1

Here, we consider the case $S = F + R > 0$, where we do not require both $F$ and $R$ to be strictly positive, instead either $F > 0$ or $R > 0$ is sufficient. Then, (34) yields the following upper bound:
\[
\sum_{t=1}^{T} f_{t}(w_{t,k}) - f_{t}(w) + r(w_{t,k}) - r(w) \\
\leq \frac{D^{2}K}{2} \left[ \frac{1}{\mu_{1}} + \sum_{t=2}^{T} \left( \frac{1}{\mu_{t}} - \frac{1}{\mu_{t-1}} - S \right) \right] \\
+ AD + 4A^{2}K \left( 1 + \frac{3\sqrt{K}}{1 - \sigma} \right) \sum_{t=1}^{T} \mu_{t}. \tag{38}
\]

Picking $\mu_{t} = (1/St)$, we obtain $(1/\mu_{t}) - (1/\mu_{t-1}) - S = 0$, which yields
\[
\sum_{t=1}^{T} f_{t}(w_{t,k}) - f_{t}(w) + r(w_{t,k}) - r(w) \\
\leq AD + 4SD^{2}K \left( 1 + \frac{3\sqrt{K}}{1 - \sigma} \right) \frac{1 + \log T}{S} \tag{39}
\]
since
\[
\sum_{t=1}^{T} \frac{1}{St} \leq \frac{1 + \log T}{S}. \tag{40}
\]
This concludes the proof of Corollary 1. \hfill \Box
C. Proof of Lemma 1

Let \( W_t \equiv [w_{t,1}, \ldots, w_{t,K}] \), \( G_t \equiv [g_{t,1}, \ldots, g_{t,K}] \), and \( H_t \equiv [h_{t,1}, \ldots, h_{t,K}] \). Then, by letting \( 1_k \) denotes the vector of all zeros but only a single one at its \( k \)th entry, we can write the desired norm as

\[
\| \bar{w}_t - w_{t,k} \| = \left\| W_t \left( \frac{1}{K} 1 - 1_k \right) \right\|. \tag{41}
\]

According to our update rule, we have the following recursion:

\[
W_t = W_{t-1}C^{t-1} - \sum_{s=1}^{t-1} \mu_{t-s}(G_{t-s} + H_{t-s+1})C^{t-s-1}. \tag{42}
\]

Putting (42) back in (41) and noting that \( C1 = 1 \) due to Assumption 3 and \( \bar{w}_1 = w_{1,k}, \forall k \in \{1, \ldots, K\} \) due to Assumption 4, we obtain

\[
\| \bar{w}_t - w_{t,k} \|
\leq \sum_{s=1}^{t-1} \mu_{t-s} \left\| (G_{t-s} + H_{t-s+1}) \left( \frac{1}{K} 1 - C^{t-s-1}1_k \right) \right\|
\leq \sum_{s=1}^{t-1} \mu_{t-s} \| G_{t-s} + H_{t-s+1} \| \| \frac{1}{K} 1 - C^{t-s-1}1_k \|. \tag{43}
\]

Here, the first term of (43) can be upper bounded as follows:

\[
\| G_{t-s} + H_{t-s+1} \| \| \frac{1}{K} 1 - C^{t-s-1}1_k \| \leq 2A\sqrt{K} \tag{44}
\]

since the norms of the subgradients are upper bounded by \( A \) due to Assumption 2.

We then consider the second term of (43). In order to derive an upper bound, we first let \( \sigma_1(A) \geq \sigma_2(A) \geq \cdots \geq \sigma_K(A) \) denote the singular values of a matrix \( A \) and \( \lambda_1(A) \geq \cdots \geq \lambda_K(A) \) denote the eigenvalues of a symmetric matrix \( A \). Since the communication graph \( C \) is a doubly stochastic matrix (by Assumption 3), we have \( \sigma_1(C) = 1 \) and \( \lambda_1(C^T C) = 1 \). Then, we let \( B \equiv (1/K)11^T \) and consider the following norm:

\[
\left\| \frac{1}{K} 1 - C^{t-s-1}1_k \right\| = \| B1_k - C^{t-s-1}1_k \|
= \| (B - C)^{t-s}1_k \|. \tag{45}
\]

where

\[
\| (B - C)^{t-s}1_k \| \leq \sigma_1(B - C) \| (B - C)^{t-s-1}1_k \| \tag{46}
\]

\( \forall t \geq 1 \). Thus, applying (46) \( s-1 \) times to (45), we obtain

\[
\left\| \frac{1}{K} 1 - C^{t-s-1}1_k \right\| \leq \sigma_2^{t-s-1}(B - C)1_k. \tag{47}
\]

Here, we observe that \( C \) is a doubly stochastic matrix by Assumption 3, hence, \( \lambda_1(C) = 1 \). Therefore, the eigenspectrums of \( B - C \) and \( (B - C)^{t-s}(B - C) \) are equal to the eigenspectrums of \( C \) and \( C^TC \), respectively, except the largest eigenvalues, i.e., \( \lambda_1(C) = \lambda_1(C^TC) = 1 \), are removed. Thus, from (47), we obtain

\[
\left\| \frac{1}{K} 1 - C^{t-s-1}1_k \right\| \leq \sigma_2^{t-s-1}(C). \tag{48}
\]

Putting (44) and (48) back in (43), we obtain

\[
\| \bar{w}_t - w_{t,k} \| \leq 2A\sqrt{K} \sum_{s=1}^{t-1} \mu_{t-s}\sigma_2^{t-s-1}(C) \tag{49}
\]

which is the desired result. This concludes the proof of Lemma 1.

\[ \square \]

V. SIMULATIONS

In this section, we evaluate the performance of distributed sequential ELM (DSS-ELM) algorithm (presented in Algorithm 1) for various multiagent networks with different regularization factors. We illustrate the superior performance of our algorithm for different regression applications involving both real and synthetic data.

A. Stationary Scenario

In this section, we consider a multiagent network of size \( K = 10 \), where each agent observes a desired signal through a linear model. In particular, the \( k \)th agent observes

\[
y_{t,k} = \psi^T x_{t,k} + n_{t,k} \tag{50}
\]

where \( \psi \in \mathbb{R}^{10} \) is chosen from a normal distribution, while 50% of the entries are randomly set to zero, \( x_{t,k} \in \mathbb{R}^{10} \) is a realization of a spatially and temporally independent normal random vector process, and \( n_{t,k} \in \mathbb{R} \) is a zero-mean white Gaussian noise with variance \( \sigma_n^2 = 10^{-3} \). We normalize the input and output attributes to the interval \([0, 1]\) as recommended in [2].

The empirical loss function is chosen as the square error loss, i.e., \( f_{t,k} = (y_{t,k} - \hat{y}_{t,k})^2 \), whereas the regularization loss function is chosen as \( \ell_1 \)-norm regularization in Fig. 2 and \( \ell_2^1 \)-norm regularization in Fig. 2, where we set \( \lambda = 10^{-4} \). We point out that for these regularization factors, the update rule (i.e., the sixth line in Algorithm 1) is given by

\[
\begin{align*}
\bar{w}_{t+1,k} &= \begin{cases}
\text{sign}(w'_{t+1,k}) \odot \left[ |w'_{t+1,k}| - \mu_t \lambda \right]_+ & \text{for } \ell_1 \\
1 + \mu_t \lambda & \text{for } \ell_2^1.
\end{cases}
\end{align*}
\tag{51}
\]
In Figs. 2 and 3, we compare the performance of the DSS-ELM algorithm with the distributed stochastic gradient (D-SG) algorithm [8], and their centralized variants, i.e., the centralized stochastic gradient (C-SG) and the centralized sequential splitting ELM (CSS-ELM) algorithms. In the distributed multiagent framework, each agent exchanges information with its neighboring agents over a communication network. We have generated this network randomly. Agents employ the uniform combination rule [38], where $C_{j,k} = 1/\pi_k$ with $\pi_k$ representing the number of neighboring agents of the $k$th agent. In the centralized framework, all the data observed by every agent are collected at a central processing unit and processed through the conventional processing techniques. In the experiments, the step sizes of the D-SG and C-SG algorithms are set to 0.05, whereas it is 0.1 for the DSS-ELM and CSS-ELM algorithms (these learning rates are chosen for a fair performance comparison, since the number of hidden nodes is approximately twice of the length of the linear parameter vector). We set the number of hidden neurons to 24 in order to achieve a reasonable steady-state loss and use a sigmoidal additive activation function. We have performed 250 independent trials (with random hidden neuron initializations) and presented the averaged results.

In Figs. 2 and 3, we observe the performance of the distributed stochastic gradient algorithm significantly outperforms the D-SG algorithm for both $\ell_1$ and $\ell_2$ regularization costs. For the $\ell_1$-regularization case, the performance of the D-SG algorithm deteriorates with respect to the $\ell_2$-regularization case. On the other hand, the performance of the proposed DSS-ELM algorithm is not significantly affected from these regularization factors, since we use the forward–backward splitting method. Furthermore, since the underlying parameter vector $\mathbf{v}$ is sparse, the linear models fail to achieve low error rates, whereas the proposed SLFN-based algorithm yields a considerably smaller error.

We next consider the following nonlinear regression model:

$$y_{t,k} = \text{sign}(\mathbf{v}^T \mathbf{x}_{t,k})$$

(52)

where $\mathbf{v}$ and $\mathbf{x}_{t,k}$ are chosen as in the previous setup, and the labels of 10% of the desired signals, i.e., $y_{t,k}$’s, are flipped in order to introduce label noise to the model. In Fig. 4, we present the performances of the DSS-ELM, CSS-ELM, D-SG, and C-SG algorithms for the $\ell_2$-regularization loss. We observe that the proposed DSS-ELM algorithm significantly outperforms the linear gradient descent-based algorithms. This follows, since the SLFN-based structures can elegantly capture the nonlinear relationships between the desired data and the regressor vectors, whereas linear models cannot learn these relationships satisfactorily. Furthermore, the proposed DSS-ELM algorithm achieves the performance of the centralized CSS-ELM algorithm, which verifies the asymptotical convergence results presented in Theorem 1 and Corollary 1.

### B. Nonstationary Scenario

In this example, we evaluate the performance of the DSS-ELM, CSS-ELM, D-SG, and C-SG algorithms in nonstationary environments. We consider the regression setup in (50), where the unknown parameter vector $\mathbf{v}$ evolves in time through a random walk model (we emphasize that this setup is extensively used in the machine learning and signal processing literature [39]) as follows:

$$v_{t+1} = v_t + q_t$$

(53)

where $q_t \in \mathbb{R}^d$ is a realization of independent identically distributed zero-mean Gaussian random vector process with autocovariance matrix $10^{-6}I_d$.

In Fig. 5, we present the time evolution of the $\ell_2$-regularized square error loss function of the proposed algorithms. Comparing Fig. 5 with the stationary setup in Fig. 3, we observe that our algorithm suffers approximately 10-dB performance loss, whereas the linear learning models suffer approximately 20-dB performance loss. Therefore, the proposed DSS-ELM algorithm is considerably more robust against the nonstationarity compared with the linear learning algorithms. This follows.
C. Real Data Sets

In this section, we illustrate the performance of our algorithm for the benchmark data sets, i.e., California housing, protein tertiary structure, and household electrical power consumption [40], [41]. In these data sets, we compare the proposed DSS-ELM algorithm with the Online Sequential (OS)-ELM algorithm of [2]. Throughout this section, we use the sigmoidal additive activation function to generate SLFNs, and we use squared error loss function (with different regularization terms for the DSS-ELM algorithm). We normalize the input and output attributes of the data set to the interval [0, 1] as recommended in [2]. We use 100 data samples (which is more than the recommended value in [2]) at the initialization phase of the OS-ELM algorithm. For the DSS-ELM algorithm, we randomly generate a distributed multiagent network with the specified size, and we set the step size \( \mu_t = 10^{-2} \) and the regularization constant \( \lambda = 10^{-4} \).

1) California Housing data set involves the prediction of the median housing value based on the information collected on several parameters through all the block groups in California from the 1990 census. The number of hidden nodes is set to 90 for the DSS-ELM algorithm and 50 for the OS-ELM algorithm. For the DSS-ELM algorithm, the number of agents is set to four for this experiment.

2) Protein Tertiary Structure data set considers the learning the size of the residue based on several other parameters, e.g., the molecular mass weighted exposed area. The number of hidden nodes is set to 50 for the DSS-ELM algorithm and 70 for the OS-ELM algorithm. For the DSS-ELM algorithm, the number of agents is set to 8 for this experiment.

3) Household Electrical Power Consumption data set contains 2075259 measurements of individual households gathered between December 2006 and November 2010, where the aim is to predict the global active power. In our experiments, we used the initial \( \sim 25\% \) of the data corresponding to the data between December 2006 and November 2007. The number of hidden nodes is set to 70 for the DSS-ELM algorithm and 50 for the OS-ELM algorithm. For the DSS-ELM algorithm, the number of agents is set to ten for this experiment.

We emphasize that the number of hidden nodes for the DSS-ELM and OS-ELM algorithms is carefully chosen to minimize the performance of the corresponding algorithms for a fair comparison between these methods. The number of agents is chosen according to the data length to prevent undertraining issues. As an example, for short data sequences, it may not be possible to train the SLFNs for huge number of agents, since each agent observes only a portion of the underlying data.

For each data set, the number of attributes as well as the length of the data that is used for training and testing can be found in Table I. We randomly choose the training and testing data for each trial and evaluate the average performance of the proposed algorithms over 50 independent trials (with random hidden neuron initializations). Note that the OS-ELM algorithm operates on a single centralized agent, whereas the DSS-ELM algorithm works for multiagent systems. Therefore, for the OS-ELM algorithm, the centralized agent is trained using all the training data, whereas for the DSS-ELM algorithm, each agent is trained using only a randomly chosen portion of the training data. As an example, for California housing data set, we have \( K = 4 \) agents and a training data of length 12000. In this scenario, the OS-ELM algorithm is trained using the entire training data, whereas for the DSS-ELM algorithm, this entire training data are randomly separated into four different training data of length 3000 and assigned to different agents.

Table II shows the training times and the Root Mean Square Error (RMSE) performances for training as well as
<table>
<thead>
<tr>
<th>Data Set</th>
<th># attributes</th>
<th># training data</th>
<th># test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>California Housing</td>
<td>8</td>
<td>12000</td>
<td>8620</td>
</tr>
<tr>
<td>Protein Tertiary</td>
<td>9</td>
<td>36000</td>
<td>9730</td>
</tr>
<tr>
<td>Household Power Consumption</td>
<td>7</td>
<td>450000</td>
<td>46067</td>
</tr>
</tbody>
</table>

**TABLE II**

**Comparison of the RMSE Performance of the Algorithms**

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Algorithms</th>
<th>Time (s)</th>
<th>RMSE (training)</th>
<th>RMSE (testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>SD</td>
</tr>
<tr>
<td>California Housing</td>
<td>DSS-ELM ($\ell_1$)</td>
<td>0.0749</td>
<td>0.1776</td>
<td>0.0011</td>
</tr>
<tr>
<td></td>
<td>DSS-ELM ($\ell_2^2$)</td>
<td>0.0705</td>
<td>0.1745</td>
<td>0.0008</td>
</tr>
<tr>
<td></td>
<td>OS-ELM</td>
<td>1.3219</td>
<td>0.1304</td>
<td>0.0014</td>
</tr>
<tr>
<td>Protein Tertiary</td>
<td>DSS-ELM ($\ell_1$)</td>
<td>0.0995</td>
<td>0.0635</td>
<td>0.0005</td>
</tr>
<tr>
<td></td>
<td>DSS-ELM ($\ell_2^2$)</td>
<td>0.1204</td>
<td>0.0633</td>
<td>0.0005</td>
</tr>
<tr>
<td></td>
<td>OS-ELM</td>
<td>5.1350</td>
<td>0.2308</td>
<td>0.0012</td>
</tr>
<tr>
<td>Household Power Consumption</td>
<td>DSS-ELM ($\ell_1$)</td>
<td>1.3709</td>
<td>0.0164</td>
<td>0.0011</td>
</tr>
<tr>
<td></td>
<td>DSS-ELM ($\ell_2^2$)</td>
<td>1.3127</td>
<td>0.0105</td>
<td>0.0009</td>
</tr>
<tr>
<td></td>
<td>OS-ELM</td>
<td>123.5087</td>
<td>0.0176</td>
<td>0.0323</td>
</tr>
</tbody>
</table>

**TABLE III**

**RMSE Performance of the DSS-ELM Algorithm for Different Network Sizes for the Protein Tertiary Data With $\ell_2^2$-Regularization**

<table>
<thead>
<tr>
<th># agents</th>
<th># samples per agent</th>
<th>Time (s)</th>
<th>RMSE (training)</th>
<th>RMSE (testing)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Mean</td>
<td>SD</td>
</tr>
<tr>
<td>4</td>
<td>9000</td>
<td>0.2619</td>
<td>0.0627</td>
<td>4.5e-4</td>
</tr>
<tr>
<td>5</td>
<td>7200</td>
<td>0.1959</td>
<td>0.0621</td>
<td>2.7e-4</td>
</tr>
<tr>
<td>6</td>
<td>6000</td>
<td>0.1634</td>
<td>0.0633</td>
<td>5.9e-4</td>
</tr>
<tr>
<td>8</td>
<td>4500</td>
<td>0.1211</td>
<td>0.0626</td>
<td>4.6e-4</td>
</tr>
<tr>
<td>9</td>
<td>4000</td>
<td>0.1072</td>
<td>0.0640</td>
<td>7.0e-4</td>
</tr>
<tr>
<td>10</td>
<td>3600</td>
<td>0.0977</td>
<td>0.0647</td>
<td>8.0e-4</td>
</tr>
<tr>
<td>12</td>
<td>3000</td>
<td>0.0827</td>
<td>0.0657</td>
<td>8.9e-4</td>
</tr>
</tbody>
</table>

Testing for the DSS-ELM and OS-ELM algorithms. Table II shows that the DSS-ELM algorithm can train the SLFNs at each agent significantly faster than the OS-ELM algorithm. As an example, the training time for the DSS-ELM algorithm (that employs $\ell_2^2$-regularization) is $\sim 1.33$ s at each agent, where we have ten agents in the distributed multiagent network. On the other hand, for the OS-ELM algorithm, the training time is $\sim 124.80$ s. Therefore, our algorithm presents an order of magnitude improvement in terms of the training time. Furthermore, even though the computational complexity of the proposed DSS-ELM algorithm is much lower than the OS-ELM algorithm, it yields a significantly better RMSE performance than the OS-ELM algorithm for protein tertiary and household power consumption data sets. This follows, since the DSS-ELM algorithm can adapt nonstationary environments in a faster manner compared with the OS-ELM algorithm. The proposed DSS-ELM algorithm uses a gradient descent-based learning algorithm that can effectively track nonstationarity (as also shown in Section V-B), whereas the OS-ELM algorithm uses a recursive least squares-based approach, which is less resilient to nonstationarity.

In Table II, we also observe that the variance of the DSS-ELM algorithm is usually smaller than the OS-ELM algorithm. The main reason behind this difference in the variances is due to the regularization constraints in the formulation of the DSS-ELM algorithm. The DSS-ELM algorithm presents a more robust performance compared with the OS-ELM algorithm due to the regularization constraints. Furthermore, the $\ell_2^2$-regularization variant of the DSS-ELM has a marginally smaller variance compared with the $\ell_1$-regularization variant. That is because, the gradient of the $\ell_1$-norm is bigger than the gradient of the $\ell_2^2$-norm when the parameter vector is close to its optimal value.
Finally, in Table III, we analyze the effects of the network size (i.e., the number of agents) to the performance of the overall system. As can be seen from Table III, as the number of agent increases, the training time of the SLFNs at each agent decreases (as the number of training data per agent decreases). On the other hand, the RMSE and the error variance at each agent increase as the number of agents increases, which is expected from the convergence upper bounds presented in Theorem 1 and Corollary 1.

VI. CONCLUSION

We study the sequential training of SLFNs over distributed multiagent networks. The aim of each agent is to minimize arbitrary cost functions by training an individual SLFN-based regressor (or classifier) using the data that is revealed only to this particular agent. On the other hand, the goal of the multagent network is to train these individual SLFNs at each agent as well as the optimal centralized batch SLFN that has access to all the data revealed to all agents. We solve this problem by introducing a distributed sequential ELM-based algorithm. In particular, we show that our algorithm guarantees that the normalized accumulated error of these individual SLFNs at each agent is asymptotically the same as the normalized accumulated error of the optimal centralized batch SLFN that is trained by an ELM-based method. The proposed algorithm works in a truly sequential manner, such that it can be used without any training or initialization phase. Furthermore, the computational complexity of the algorithm is only linear in the number of hidden nodes; hence, it is highly appealing for applications involving big data.

REFERENCES


N. Denizcan Vanlı received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree in electrical engineering and computer science with the Massachusetts Institute of Technology, Cambridge, MA, USA. His current research interests include convex optimization, online learning, and distributed optimization.

Muhammed O. Sayın received the B.S. and M.S. degrees in electrical and electronics engineering from Bilkent University, Ankara, Turkey, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Illinois at Urbana–Champaign, Champaign, IL, USA. His current research interests include distributed networks, optimal control, and dynamic games.

Ibrahim Delibaltu received the B.S. degree in electrical engineering from Boğaziçi University, Istanbul, Turkey, in 1993, and the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 1995. He is currently pursuing the Ph.D. degree in design, technology, and society, an interdisciplinary program, with Koç University, Istanbul, with a focus on social sciences and machine learning. He was with Intel, Santa Clara, CA, USA, Silicon Graphics, Milpitas, CA, USA, and Cisco Systems, San Jose, CA, USA, where he was involved in the field of high performance microprocessor and network processor chip design from 1995 to 2010. He has held various positions with the Türk Telekom Group, Ankara, Turkey, since 2011, where he is currently leading the Emerging Services and Research and Development Team.

Suleyman Serdar Kozat (A’10–M’11–SM’11) received the B.S. (Hons.) degree from Bilkent University, Ankara, Turkey, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Urbana, IL, USA.

He joined the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, as a Research Staff Member and later became a Project Leader with the Pervasive Speech Technologies Group, where he focused on problems related to statistical signal processing and machine learning. He was a Research Associate with the Cryptography and Anti-Piracy Group, Microsoft Research, Redmond, WA, USA. He is currently an Associate Professor with the Electrical and Electronics Engineering Department, Bilkent University. He has co-authored over 100 papers in refereed high impact journals and conference proceedings and holds several patent inventions (currently used in several different Microsoft and IBM products, such as MSN and ViaVoice). He holds several patent inventions due to his research accomplishments with the IBM Thomas J. Watson Research Center and Microsoft Research. His current research interests include cyber security, anomaly detection, big data, data intelligence, adaptive filtering, and machine learning algorithms for signal processing.

Dr. Kozat received many international and national awards. He is the Elected President of the IEEE Signal Processing Society, Turkey Chapter.